

**Administração de sistemas**



**Pulumi**

**José Nestor, Renaldo Santino, Guilherme Araujo  
e Henrique Vitor**



# Sumário

Pulumi



Introdução

Descrição da ferramenta

Vantagens e limitações

Integrações

Instalação e configuração

Exemplo prático

# Introdução

Infrastructure as Code (IaC):  
Resolvendo desafios da administração de  
Infraestrutura.



# Problemas

Dificuldades da  
administração de  
infraestrutura manual



Gerenciamento manual propenso a falhas.

Ambientes inconsistentes entre dev, staging e produção.

Dificuldade de escalar rapidamente.

Rastreabilidade e versionamento inexistentes.

# Como o IaC resolve esses problemas?

Descrição da infraestrutura usando código



**Automação:** elimina tarefas repetitivas.



**Consistência:** ambientes idênticos (dev, staging, produção).



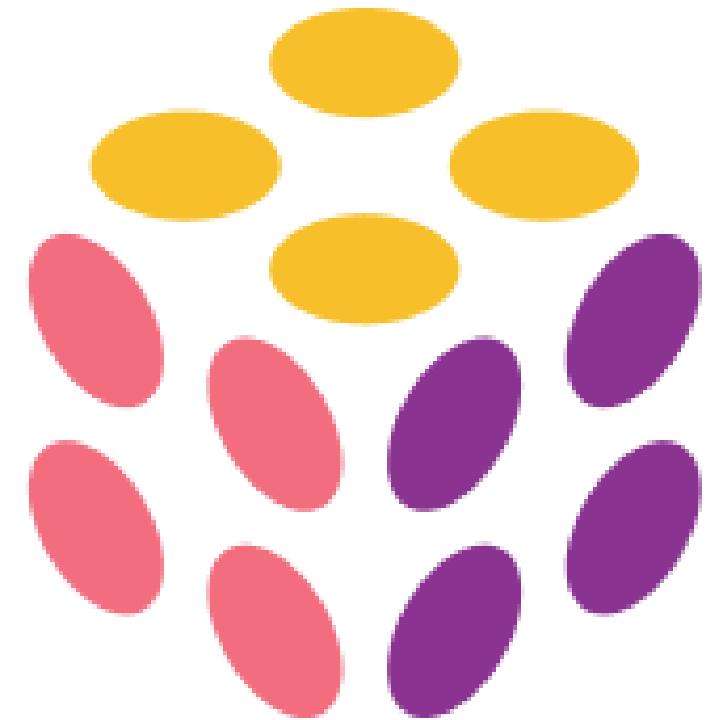
**Escalabilidade:** provisionamento rápido em nuvem.



**Versionamento:** código versionado no git.



**Segurança:** políticas aplicadas via código.



# Pulumi

## Descrição da ferramenta

Definição, propósito e principais funcionalidades.

# Pulumi?

- Pulumi é uma ferramenta de infraestrutura como código (IaC) que permite criar, configurar e gerenciar recursos de nuvem usando linguagens de programação comuns como Python, JavaScript, Go e C#.

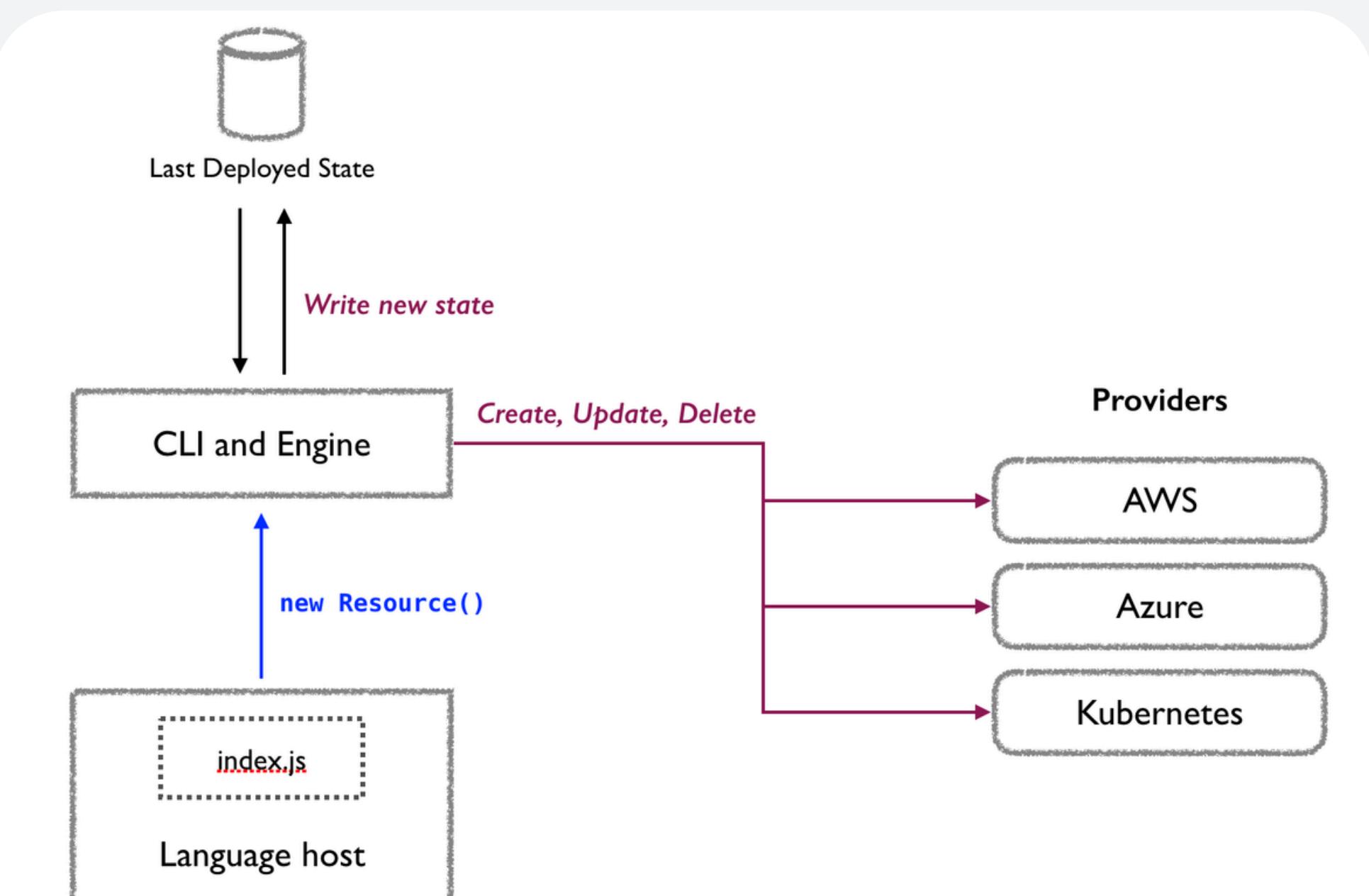
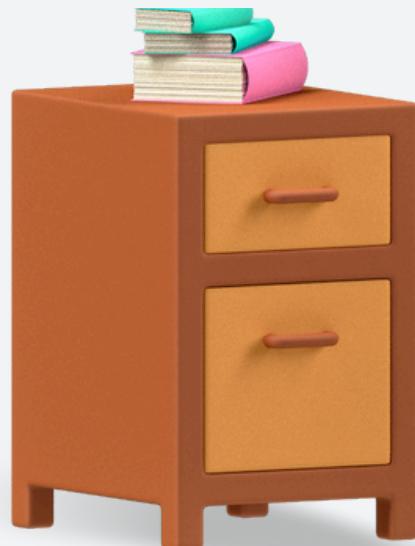
# Como funciona?

- Modelo declarativo baseado em estado desejado.
- Infraestrutura escrita com linguagens reais.
- Integração com IDEs, autocomplete, testes, etc.

# Arquitetura do Pulumi

Principais componentes:

- Language Host
- Deployment Engine
- Resource Providers



# Exemplo de código

```
from pulumi_aws import s3

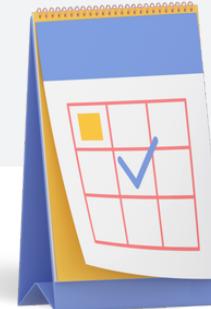
media_bucket = s3.BucketV2('media-bucket')
content_bucket = s3.BucketV2('content-bucket')
```

Python

- **pulumi up** inicia a execução.
- Language Host registra os recursos desejados.
- Engine orquestra a criação/atualização/deleção.
- Providers interagem com nuvem (ex: AWS).

# Atualizações, Deleções e Gerenciamento de Dependências

Como o Pulumi aplica mudanças com segurança?



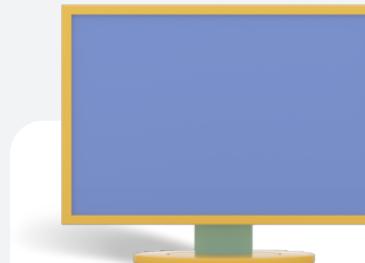
## DETECÇÃO AUTOMÁTICA DE MUDANÇAS

O Pulumi compara o estado atual com o desejado e gera um plano de execução. Recursos não mais registrados no código são removidos da infraestrutura.



## ATUALIZAÇÕES INCREMENTAIS E INTELIGENTES

Apenas os recursos que mudaram são afetados (evita recriação desnecessária).



## MANTÉM ORDEM E PARALELISMO COM SEGURANÇA

Entende as relações entre os recursos e, sempre que possível, os cria simultaneamente para melhorar a performance.

# Outras Características

## REUTILIZAÇÃO E MODULARIZAÇÃO

Você pode modularizar seu código em funções, pacotes, classes... como num app normal.

## PREVIEW E CONTROLE DE MUDANÇAS

Com `pulumi preview`, veja exatamente o que será criado, alterado ou removido antes de aplicar mudanças.

## SUPORTE A TESTES E TIPAGEM

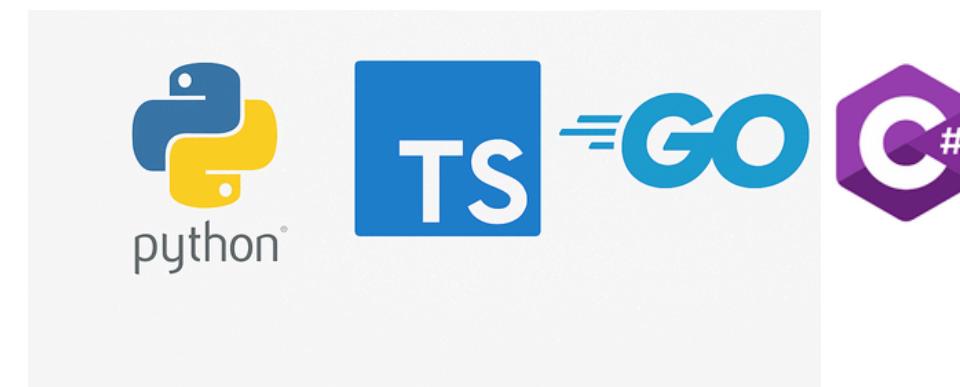
Por ser código real, é possível escrever testes unitários e mocks com frameworks da linguagem, além de usar tipagem forte (Ex: Java).

O Pulumi junta o código com o controle da infraestrutura. É uma ferramenta pensada para desenvolvedores que querem produtividade sem abrir mão de boas práticas.

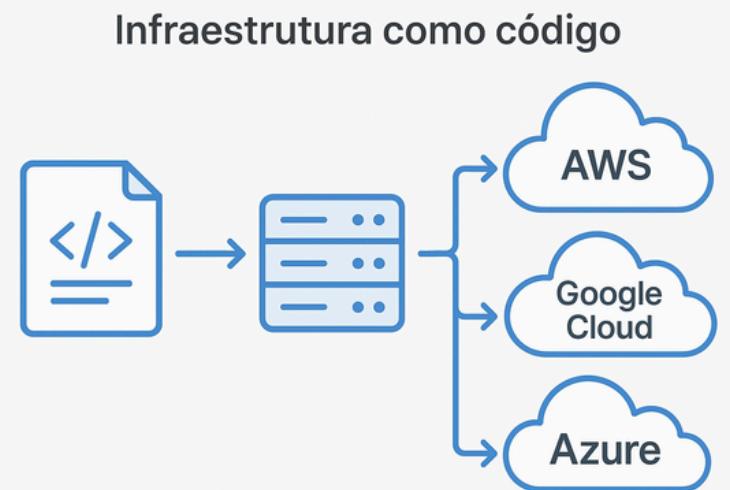


# Vantagens e limitações

Benefícios e desafios ao usar a ferramenta.



# Perguntas relevantes



Quais as vantagens do Pulumi?

Quais as desvantagens?

Quando vale a pena usar?

Quantos aqui já usaram Terraform? E quem já experimentou Pulumi?"



# Quais as vantagens?



- **Linguagens familiares.** (Python, TypeScript, JavaScript, Go, C#, F#, Java, YAML.)
  - Converte HCL para Pulumi
- **Suporte IDE**
- **Reutilização de código.** (loops, funções, bibliotecas)
- **Multi-cloud.** (AWS, Azure, GCP, Kubernetes)
- **Estado gerenciado.**
- **Provedores nativos** para AWS, Azure, Google Cloud e Kubernetes.
- **Teste e Validação.** (unitários, integração e propriedade)
- **Gestão de Segredos**



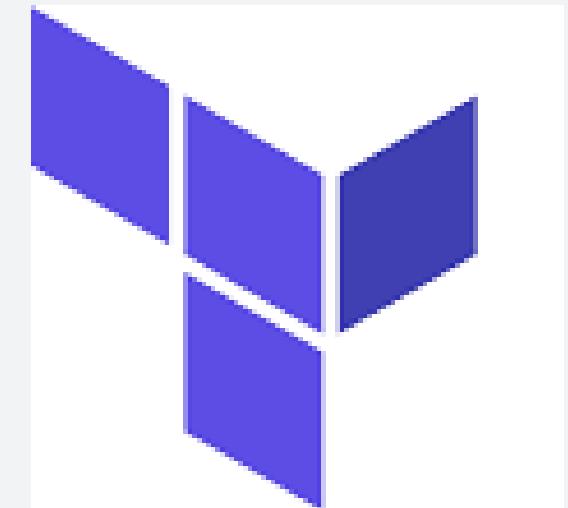
# Quais as desvantagens?



- **Maturidade vs. Terraform.** "Menos módulos prontos. Suporte limitado."
- **Dependência de SDKs.** "Novos recursos de cloud podem demorar a chegar nos SDKs do Pulumi."
- **Complexidade de Debug.** Erros em Python/TypeScript são menos intuitivos que HCL para problemas de infraestrutura."
- **Custo em Escala.** "Recursos avançados exigem planos pagos."



# Pulumi x Terraform



| Critério   | Pulumi              | Terraform        |
|------------|---------------------|------------------|
| Linguagem  | Python/TS/Go        | HCL              |
| Estado     | Gerenciado ou local | Auto-gerido (S3) |
| Comunidade | Crescente           | Madura           |

# Quando usar Pulumi?

## Use Pulumi se:

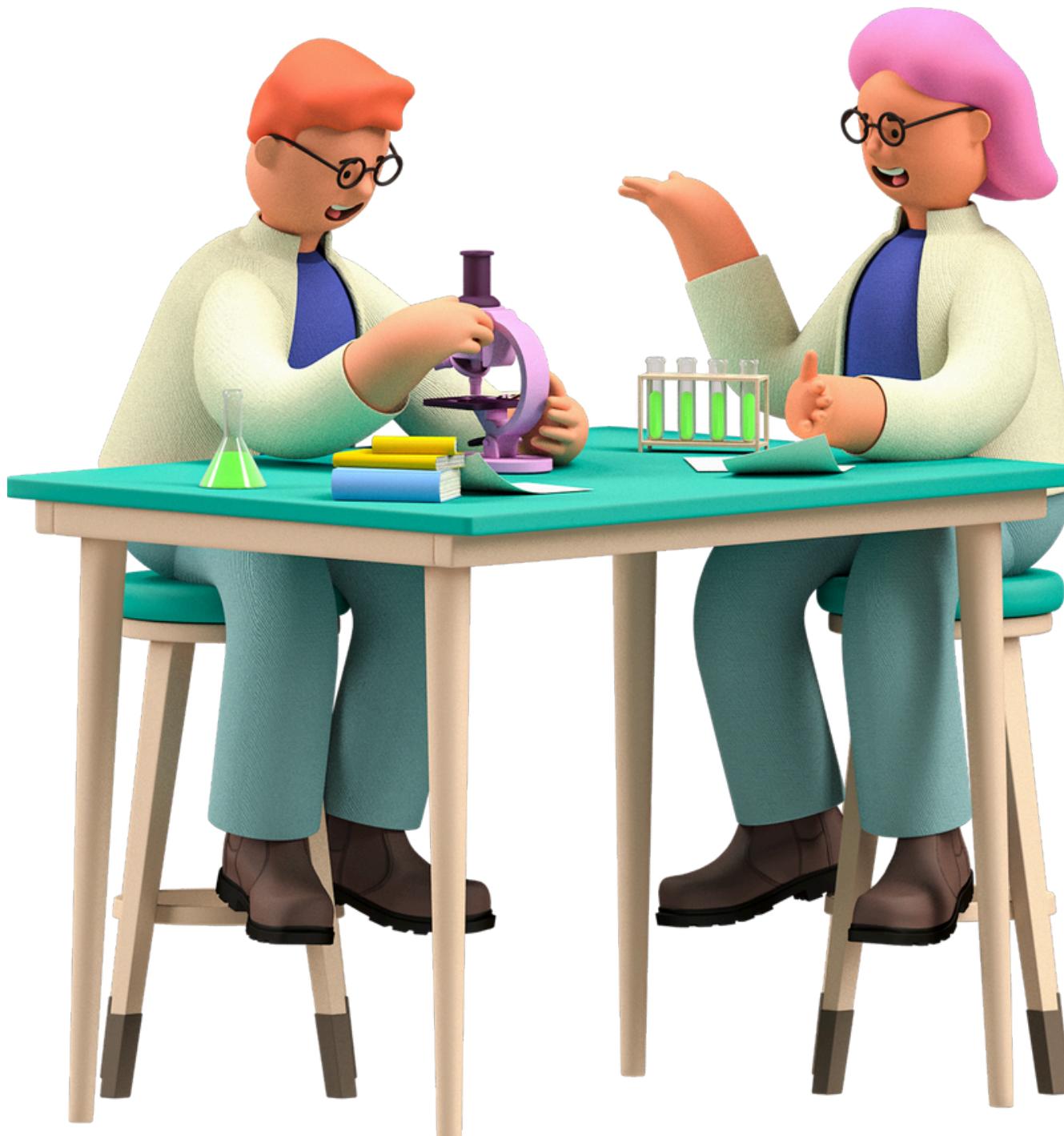
- Sua equipe já domina Python/TypeScript/Go.
- Você quer reutilizar lógica programática (loops, funções).
- Precisa de multi-cloud (AWS + Azure + K8s).

## Evite Pulumi se:

- Você depende de módulos Terraform muito específicos.
- Resistência da equipe por já estar familiarizada com HCL.
- Necessidade de algum suporte que o Pulumi não oferece.

# Integrações

Interações com outras soluções no ecossistema de administração de sistemas.



# Pulumi + Provedores de Nuvem

- SDKs oficiais para integração com os principais provedores de nuvem
  - AWS, Azure, GCP, etc
- Permite criar e gerenciar recursos de infraestrutura diretamente no código
- Provisionamento Automatizado
- Reutilização de código em ambientes diferentes



# Pulumi + Kubernetes

- Gerenciamento tanto do cluster quanto dos recursos dentro dele
  - Criação de clusters
  - Definição de objetos: Deployment, Service, ConfigMap, etc.
- Unificação de infraestrutura e deployment em um único projeto
- Definição de recursos direto no código, sem o uso de arquivos YAML
- Vantagens oferecidas por linguagens de programação
  - Loops, reuso de código em funções, classes, etc



kubernetes

# Pulumi + Docker

- Automatiza de processos
  - Criação de imagens a partir de Dockerfiles
  - Push para registries
  - Rodar aplicações em containers na nuvem
- Elimina a necessidade de arquivos de configuração separados
- Tudo pode ser descrito em uma única stack Pulumi



# Pulumi + Monitoramento

- Integração com serviços de monitoramento por meio de SDKs ou APIs
  - Prometheus, CloudWatch, etc
- Automação de criação e configuração de alertas, dashboards e métricas
- Alertas, dashboards e métricas definidos em código versionável e reusável
- Evita configurações manuais repetitivas e aumenta a agilidade de provisionamento

# Instalação e configuração

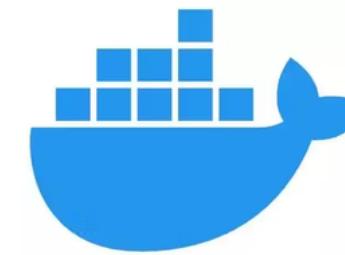
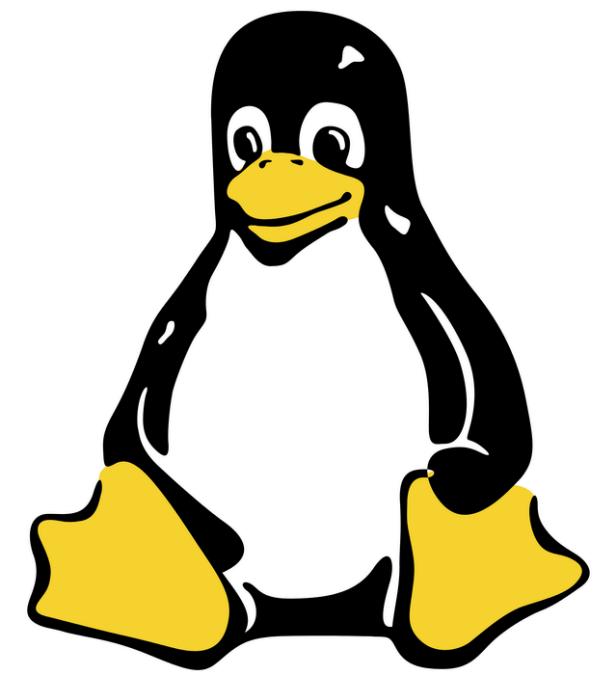
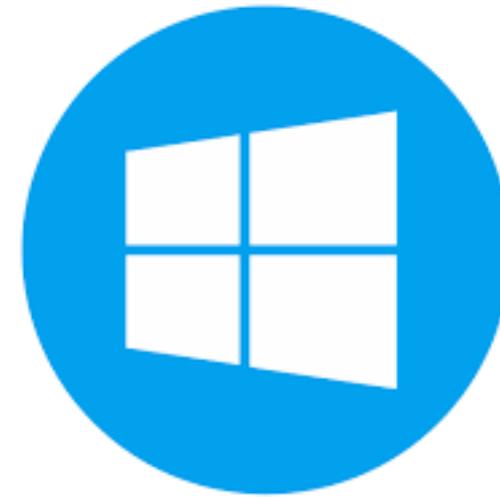
Como preparar o ambiente para utilizar o Pulumi?



# Formas de Instalação



Mac<sup>TM</sup> OS



docker

## **Linux/macOS:**

- **via script**

```
curl -fsSL https://get.pulumi.com | sh
```

- **via Homebrew**

```
brew install pulumi
```

## **Windows:**

- **Via instalador .msi Baixar em: <https://www.pulumi.com/docs/install/>**

## **Docker:**

- **docker run --rm -it pulumi/pulumi**

# Verificação da Instalação

**pulumi version**

# Criando um Projeto

**pulumi new aws-javascript**

# Configurando Provedor AWS

1. Criar chaves de acesso na AWS

2. Instalar a AWS CLI:

- curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"
- unzip awscliv2.zip
- sudo ./aws/install

3. Configure a AWS CLI no terminal:

- aws configure
- Preencha com as chaves geradas no console AWS:

AWS Access Key ID: SUA\_ACCESS\_KEY

AWS Secret Access Key: SUA\_SECRET\_KEY

Região padrão: us-east-1

Formato de saída: json



# Exemplo prático

Caso de uso prático da ferramenta em um cenário real de administração de sistemas.

# Obrigado!

